VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

BAKALÁŘSKÁ PRÁCE

2011 Miroslav Matoušek

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Ovládání grafického displeje EL16032a pomocí linky RS232

Control Grafic Display EL16032A via RS232

2011 Miroslav Matoušek

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne

Podpis studenta

Poděkování

Děkuji vedoucímu Ing. Davidu Seidlovi nejen za výbornou pedagogickou a odbornou pomoc, ale také za mnohé cenné rady při zpracování mé bakalářské práce.

Abstrakt

Tato bakalářská práce se zabývá obsluhou grafického LCD displeje EL16032a za pomocí mikročipu ATtiny2313, od firmy ATMEL. Dalším úkolem je vytvoření aplikace pro operační systém Linux. Tato aplikace komunikuje s mikročipem prostřednictvím sériového rozhraní RS232. Pomocí této aplikace je možné na displeji zobrazovat jednoduché grafické prvky.

Při práci bylo vycházeno z aplikace od Ing. Davida Seidla, která umožňovala komunikaci s displejem v textovém režimu. Tato aplikace byla navržena pro 4bitový paralelní režim. Pro grafický režim bylo nutné zvolit 8bitový paralelní režim, jelikož v 4bitovém korektně nefungoval. Pro sériovou komunikaci byla využita funkce mikročipu UART a také integrovaný obvod MAX232 pro převod napětí. Aplikace pro sériový port využívá knihovnu termios.h, která umožňuje potřebné nastavení sériového portu. Aplikace pro operační systém Linux i program pro mikroprocesor byly programovány v jazyce C.

Výsledný program pro mikročip je schopný komunikovat s LCD displejem jak v textovém tak i v grafickém režimu. Původní textový režim byl rozšířen o načítání dat z displeje, grafický režim obsahuje funkci pro vykreslení jednoho bodu. Pomocí této metody lze přes obslužnou aplikaci vykreslit jednoduché grafické prvky.

Klíčová slova:

LCD, grafický display, mikroprocesor, programování, komunikace sériovým rozhraním

Abstract

This bachelor labour presents dealing operation of graphic LCD display controlled by microchip ATtiny2313 from ATMEL. Another task is to create application for Linux operating system. This application communicates with the microchip via the serial interface RS232. With this application is possible to show simple graphic elements on the display.

This work was built on the application from Ing. David Seidl, that allows communication with the display in the text mode. This application was designed for 4-bit parallel mode. For graphics mode it was necessary to select 8-bit parallel mode, because the 4-bit did not work correctly. For serial communication was used function of microchip UART and also integrated circuit MAX232 for voltage conversion. Application for the serial port uses termios.h library, which allows the required serial port settings. Applications for the Linux operating system and program for the microprocessor was programmed in C.

The final program for the microchip is able to communicate with the LCD display both in text and in graphics mode. Original text mode has been extended by loading data from the display, graphics mode includes function to draw one point.By use this method is possible to draw a simple graphical elements through the servicing application.

Key words:

LCD, graphic display, microprocessor, programming, communications via serial interface

Seznam použitých symbolů a zkratek

LCD	Liquid crystal display
USART	Universal asynchronous transmitter
CISC	Complex Instruction Set Computer
EEPROM	Electrically Erasable Programmable Read-Only Memory
SRAM	Static Random Access Memory
I/O	input/output
CPU	Central Processing Unit
GND	Ground (electricity)
RAM	Random access Memory
OS	Operating Systém

Obsah

1. ÚVOD

Při práci s grafickými LCD displeji je velmi nutné dobře zvážit typ mikroprocesoru, který bude LCD displej řídit. Je potřeba dbát na to, aby měl dostatečný počet I/O pinů, přes které bude komunikace probíhat. Vhodný mikročip je vybrán v závislosti na LCD displeji. Inteligentní LCD displeje často umožňují různé druhy komunikace, mezi které patří sériová, 4bitová paralelní nebo 8bitová paralelní. Ovšem ne každá tato komunikace umožňuje displej naplno využívat, proto je vždy nutné pečlivě prostudovat dokumentaci displeje. Dalším problémem, který může nastat je samotné naprogramování mikročipu, pro tento účel je vhodný kompilátor AVR-DUDE, který je k dispozici zdarma. Grafické LCD displeje i mikročipy mají stanovené napájecí napětí na 5V. To může být problém při sériové komunikaci například s osobním počítačem, standardní sériové rozhraní RS232 pracuje při napětí +12V a -12V. Z tohoto důvodu je nutné využití převodníku napětí, například integrovaného obvodu MAX232.

Pro samotný program mikročipu je programátor omezen jen na některé programovací jazyky, z nichž jsou to například jazyk C, Assembler nebo Delphi. Při programování je nutno brát v potaz fakt, že mikročip obsahuje jen velmi malé množství programovatelné paměti; pokud je potřeba, můžeme ji u některých mikročipů rozšířit externí pamětí.

U sériové komunikace mikročipu a PC, je třeba dbát na některá základní nastavení, jako je například velikost přenášeného znaku (např. 8 nebo 9 bitů). Toto nastavení musí být stejné na obou stranách. Další důležitou věcí je frekvence mikročipu. Nejvhodnějším řešením je připojení externího krystalu na vývody mikročipu XTAL1 a XTAL2. Ačkoliv některé mikročipy mají již vnitřní RC člen a tudíž pro samotný běh mikročipu není externí krystal potřeba, nemusí frekvence tohoto RC členu pro sériovou komunikaci vyhovovat. Proto i v tomto případě raději RC člen, za pomocí přenastavení hodnoty FLASH FUSE BITs vypneme a využíváme již zmíněný externí krystal, například o frekvenci 9,216MHz.

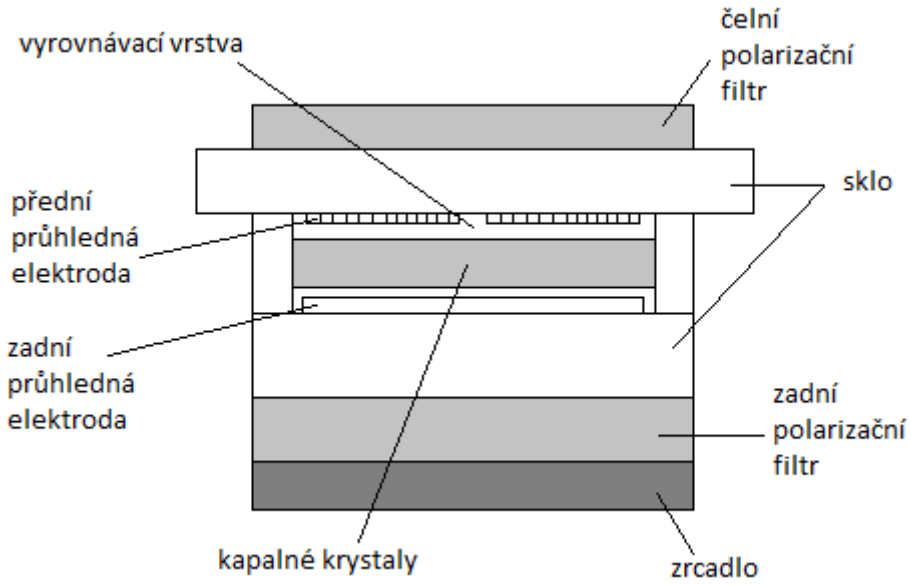
2. TEORETICKÁ ČÁST

2.1 LCD displej

LCD displeje jsou založené, jak již název napovídá, na technologii tekutých krystalů (LCD – Liquid Crystal Display). Využití tekutých krystalů spočívá v tom, že se mohou nacházet nejen v tekutém nebo pevném stavu, ale také v tak zvaném tekutém krystalickém stavu. V tomto stavu jsou krystaly charakteristické tím, že i když jsou tekuté, mají vlastnosti pevných látek.

U technologie tekutých krystalů jsou nejdůležitější dva jevy. Prvním z nich je to, že při změně polohy jejich orientovaných úhlů dochází ke změně polarizace světla, které tekutými krystaly prochází, to se na venek projeví zčernáním. Druhým jevem je chování molekul tekutých krystalů. Ty v elektrickém poli tvoří dipóly, což znamená, že jedna strana má kladný a druhá strana záporný elektrický náboj.

Natáčení dipólů v LCD displeji řídíme pomocí tranzistorů, které se starají o každý jednotlivý pixel LCD (zobrazovací bod).



Obr. 1. LCD displej – základní uspořádání [1]

Při řízení LCD displeje lze využít jak stejnosměrného, tak i střídavého napětí. Ovšem využití stejnosměrného napětí se z důvodů neblahého vlivu, elektrochemického rozkladu kapalného krystalu, nedoporučuje. Proto se vždy využívá střídavé napětí. Napětí pro ovládání LCD displejů se pohybuje v rozmezí od 10 do 20V.

Z výše uvedeného je patrné, že se samotný LCD displej ovládá poměrně problematicky. Proto se většinou setkáme s tím, že je samotný LCD displej osazen v tzv. LCD modulu.

Takovéto moduly v sobě kromě displeje obsahují řadič, zajišťující jednodušší připojení k mikrokontroleru. Dále násobič napětí, který dodává LCD displeji požadovanou úroveň napětí.

2.1.1 LCD displej EL16032a

Tento grafický LCD displej umožňuje práci ve dvou režimech: grafickém a textovém. V textovém režimu má řadič tohoto displeje přednastavenou znakovou sadu. Tato sada obsahuje písmena abecedy, číslice a čínské znaky. Abychom zobrazili libovolný znak z textové sady, stačí nám displeji zaslat pouze dvě osmibitové instrukce. První instrukcí nastavíme pozici kurzoru displeje a druhou zvolíme požadovaný znak.

V grafickém režimu sice nemáme přednastavenu znakovou sadu, ale místo toho nám řadič displeje umožňuje nezávisle adresovat kterýkoliv pixel. Nevýhodou je, že pokud v tomto režimu chceme na displeji zobrazit například písmeno „A“, musí se o jeho vykreslení postarat mikročip.

Displej podporuje v textovém režimu zobrazení až 20 znaků ve dvou řádcích, v grafickém režimu obsahuje 160x32 viditelných bodů. S tímto grafickým LCD displejem můžeme komunikovat třemi způsoby: 4bitovým paralelním, 8bitovým paralelním a sériovým. Z úsporného hlediska je nejlepší paralelní komunikace, jelikož pro samotnou komunikaci není zapotřebí využít více než pouhé tři piny jak na mikročipu, tak i displeji. Ovšem její velkou nevýhodou je, že v grafickém režimu neumožňuje čtení dat z displeje.

Pro propojení s mikročipem displej obsahuje 14 pinů. Z těchto čtrnácti pinů slouží tři jako napájecí, zbývající piny jsou datové, viz tabulka 1.

NO.	SYMBOL	FUNCTION
1	VDD	Logic power supply (5V)
2	VSS	Power supply (0V, GND)
3	VO	Power supply for LCD drive
4	RS(CS*)	Register select/adress counter read (Chip select, for seriál mod)
5	RW(SID*)	Read/Write select signal (seriál data input)
6	E(SCLK*)	Enable trigger (seriál clock)
8-14	DB0 – DB7	Data bus line

Tabulka 1. LCD displej – zapojení pinů

VDD napájecí napětí řadiče displeje
VSS zemní vodič
VO napájecí napětí pro zobrazovací jednotku
RS výběr přenosu dat nebo řídící instrukce
CS aktivace modulu (sériový mód)

RW	signál na tento pin specifikuje za se jedná o čtení nebo zápis dat
SID	pin sloužící k zasílaní dat/instrukcí (sériový mód)
E	povolovací vstup
SCLK	hodinový signál (sériový mód)
DB0 – DB7	piny sloužící k zasílání dat/instrukcí

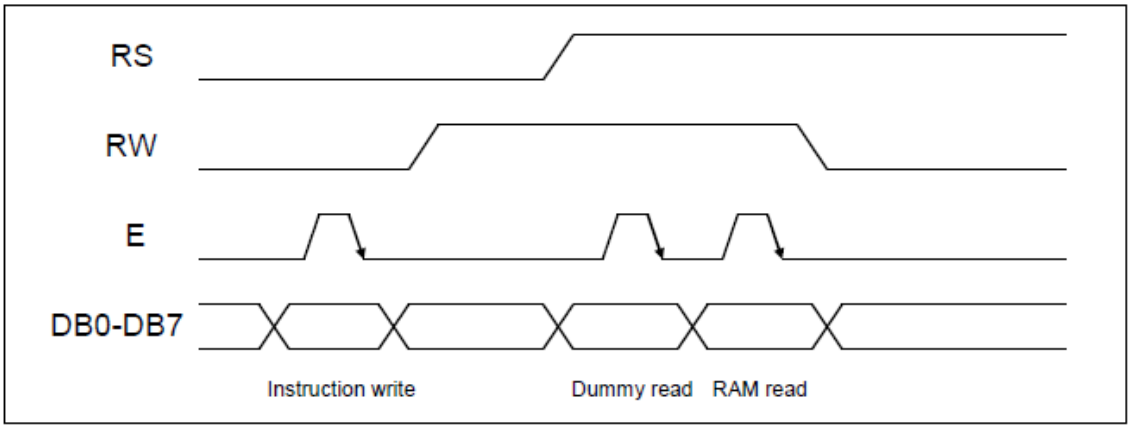
2.1.2 Podrobný popis komunikace LCD s mikročipem

Jak již bylo řečeno, s LCD displejem EL16023a lze komunikovat trojím způsobem. U čtyř a osmibitové komunikace se využívá řídících linek RS, RW a E. Pomocí linky RS rozlišujeme, zda má posílaná data displej zpracovat jako instrukci (log. 0) nebo jako data určená k zobrazení (log. 1). Linkou RW říkáme displeji, jestli chceme data číst (log. 1) anebo zapisovat (log. 0) a linkou E povolujeme přístup. Dále jsou také důležité linky DB0 až DB7, na které přivádíme data. V případě čtyřbitové komunikace si vystačíme pouze s linkami DB4 až DB7, ovšem data je potřeba zapsat nadvakrát.

U sériového přenosu využíváme linek CS, SCLK a SID. Linka CS nám podobně jako linka E v paralelním přenosu umožňuje přístup, linka SCLK slouží jako hodinový signál a pomocí linky SID přenášíme jednotlivá data.

▪ 8bitová komunikace

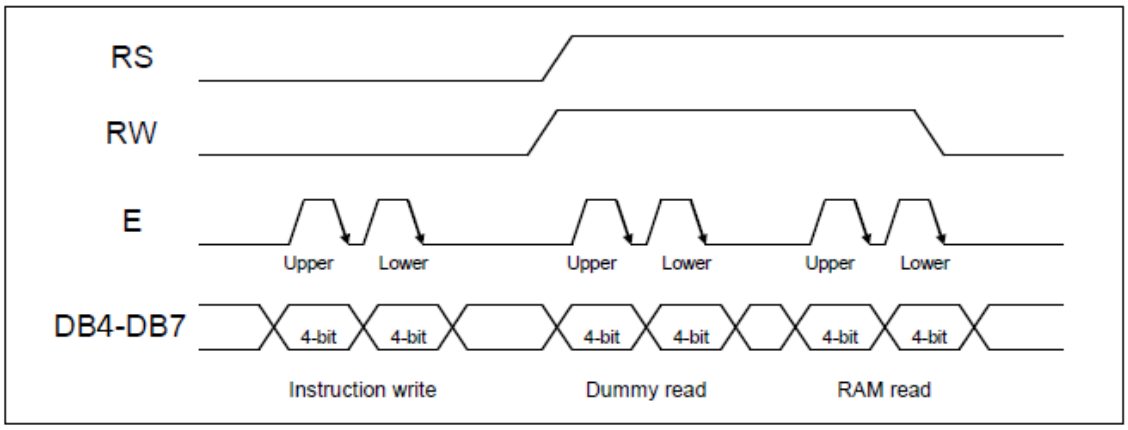
Zápis dat probíhá tím způsobem, že se nejdříve vyšle signál RS, který displeji řekne, zda se jedná o data nebo instrukci. Dále se vyšle signál RW (čtení/zápis). Po těchto dvou signálech se přivedou požadovaná data na linky DB0 až DB7 a aktivuje se přístup pomocí signálu E (log. 1), následuje časová prodleva asi 4ms než displej data přečte a opět se nastaví linka E na logickou nulu.



Obr. 2. Diagram osmibitového přenosu [2]

▪ **4bitová komunikace**

Je velmi podobná komunikaci osmibitové; jediný rozdíl je v tom, že pro posílání instrukcí a dat se využívají pouze linky DB4 až DB7, tudíž je potřeba po nastavení signálu RS a RW odeslat nejprve horní 4 bity E(1) → delay → E(0), pak spodní čtyři bity a opět aktivovat a deaktivovat linku E stejně jako u horních čtyř bitů.

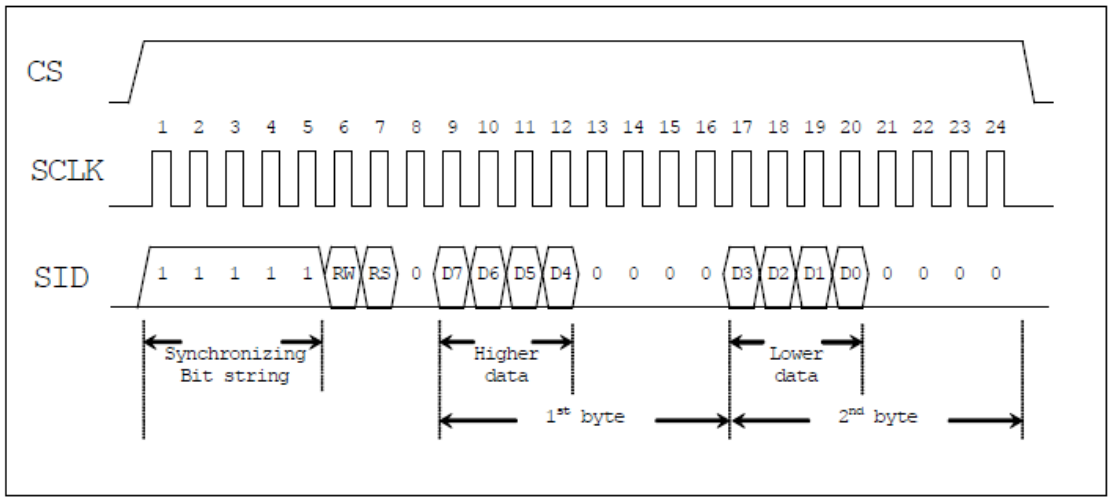


Obr. 3. Diagram čtyřbitového přenosu [2]

▪ **Sériová komunikace**

U sériové komunikace se musí nejprve aktivovat signál na lince CS (log. 1). Poté se už jen spíná hodinový signál a zasílají data po lince SID. Při každém sepnutí hodinového signálu se přijme právě jeden posílaný bit. Z obrázku číslo 4 je patrné, že se data odesílají následujícím způsobem:

1. Odešleme pět synchronizačních bitů v log. 1
2. Odešleme signály na linkách RW a RS dle potřeby buď log. 1 nebo log. 0
3. Tuto hlavičku oddělíme jednou log. nulou
4. Odešleme horní 4 bity a oddělíme je čtyřmi log. nulami
5. Odešleme spodní 4 bity a opět oddělíme čtyřmi log. nulami



Obr. 4. Diagram sériového přenosu [2]

2.1.3 Inicializace displeje

Pokud chceme zobrazovat data na displeji, je potřeba jej nejprve správně inicializovat. Až po této inicializaci se mohou na displeji zobrazovat požadovaná data.

Při inicializaci displeje se musí myslet na to, v jakém režimu a jakým druhem komunikace budeme displej ovládat. Pro každý typ komunikace se inicializace nepatrně liší. Inicializace, jak již bylo řečeno, se také liší v tom, jestli chceme pracovat v režimu grafickém nebo textovém. Toto ovlivníme hodnotou „Nastavení režimu“ viz tabulka 2

Příkaz/data	**R/W**	**RS**	**Data**								**Čas provedení**
			7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
Vymaž displej	0	0	0	0	0	0	0	0	0	1	2.5ms
Návrat na začátek	0	0	0	0	0	0	0	0	1	X	2.5ms
Volba režimu	0	0	0	0	0	0	0	1	I/D	S	60µs
Zapni/vypni displej	0	0	0	0	0	0	1	D	C	B	60µs
Posun zobrazení kurzoru	0	0	0	0	0	1	S/C	R/L	X	X	60µs
Nastavení režimu	0	0	0	0	1	DL	N	RE	G	X	60µs
Nastaveníadresy CG RAM	0	0	0	1	Adresa CG RAM						60µs
Nastavení adresy DD RAM	0	0	1	Adresa DD RAM							60µs
Čtení stavu displeje	1	0	BF	Obsah adresářového čítače							60µs
Zápis dat do RAM	0	1	Data pro CG/DD/grafickou RAM								60µs
Čtení dat z RAM	1	1	Data z CG/DD/grafickou RAM								60µs
Nastavení adr, grafické RAM	0	0	Adresa řádku/sloupce								60µs

Tabulka 2. Příkazy řadiče pro LCD displej EL16032a [1]

X libovolná hodnota (0 nebo 1)

I/D inkrementace (1), dekrementace (0)

S režim displeje (S = 0 – normální práce, S = 1 – kombinovaný posun displeje, jsou-li data zapsána)

D displej zapnut (1), vypnut (0)

C zobrazení kurzoru zapnuto (1), vypnuto (0)

S/C posun displeje, jsou-li data zapsána (1), posun kurzoru, jsou-li data zapsána (0)

R/L posun doprava (1), posun doleva (0)

DL 8bitová komunikace (1), 4bitová komunikace (0)

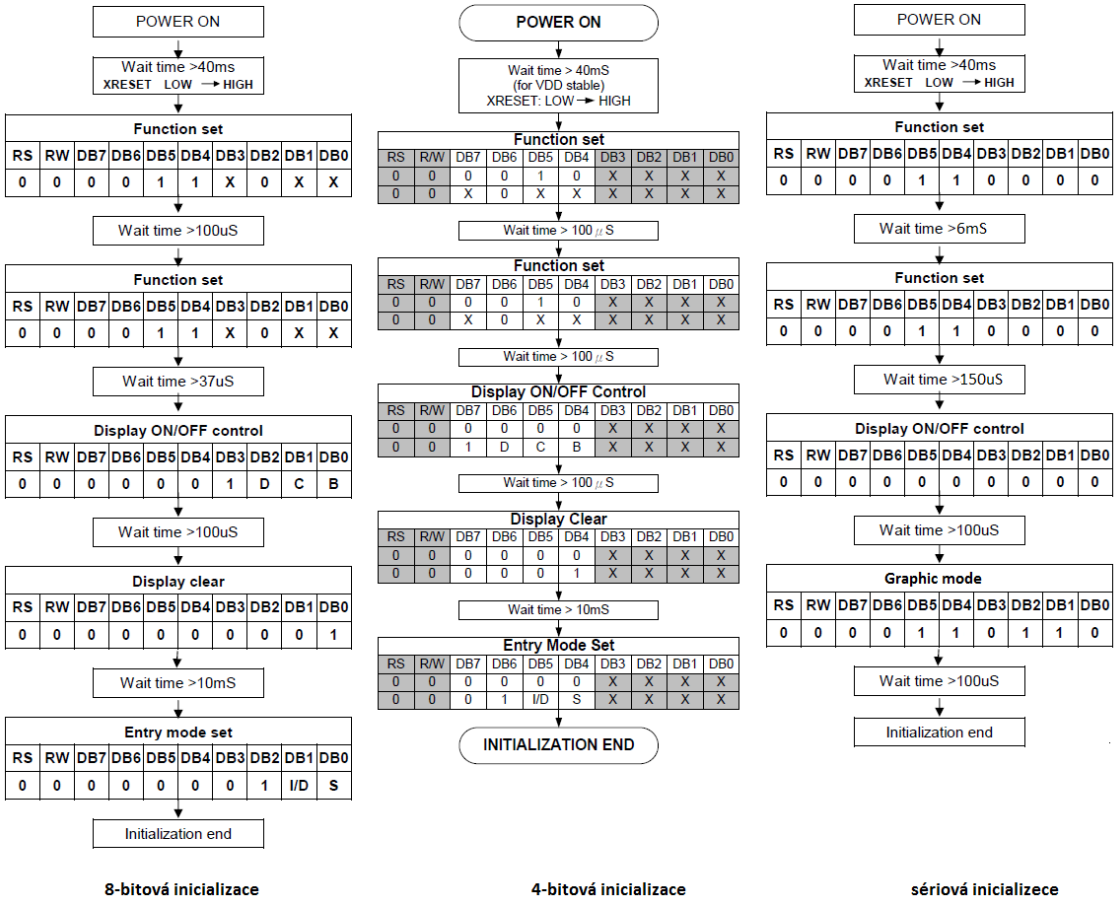
N dva řádky (1), jeden řádek (0)

RE normální příkazová sada (0), rozšířená příkazová sada (0)

G textový režim (0), grafický režim (1)

BF příznak zaneprázdnění displeje (1), připravenosti displeje (0)

Na obrázku číslo 5 je znázorněn sled po sobě jdoucích instrukcí, zasílaných mikročipem LCD displeji při různých módech komunikace. Šedá pole u 4bitové komunikace představují „nevyužití“ daných pinů.



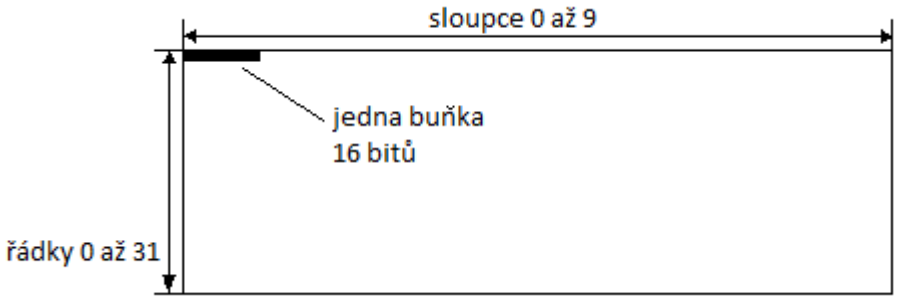
Obr. 5. Inicializace LCD displeje pro různé typy komunikace [2]

2.1.4 Komunikace LCD a mikročipu v grafickém režimu

V grafickém režimu se musí adresa i zobrazovaná data zapisovat oproti textovému režimu nadvakrát (v případě čtyřbitové komunikace na čtyřikrát). Je to z toho důvodu, že velikost jedné buňky činí 16 bitů a my jsme schopni přenést najednou maximálně osm bitů. Při zápisu dat v grafickém režimu postupujeme v tomto pořadí:

1. Nastavíme adresu v sloupci (0 až 31)
2. Nastavíme adresu v řádku (0 až 9)
3. Zašleme horní bajt a následně spodní bajt

Při adresování grafické paměti se musí myslet na to, že adresa začíná místo na nule na hodnotě 0 × 80 a to jak u sloupce, tak i u řádku. Také musí být zvolena rozšířená grafická sada i grafický režim.



Obr. 6. Adresování grafické paměti [1]

2.2 Jednočipové mikropočítače

Jednočipový mikropočítač nebo také mikrokontrolér je integrovaný elektrický obvod, který obsahuje vše potřebné k běhu interního programu, aniž by bylo potřeba připojení dalších podpůrných obvodů.

Ačkoliv se jednotlivé typy mikropočítačů ve své architektuře liší, tak jako základní části je možno uvést:

- Procesor – jehož taktovací frekvence dosahuje až stovek MHz
- Operační paměť – tato paměť je typu RAM a její velikost se pohybuje od jednotek byte až po desítky KB
- Paměť programu – typy těchto pamětí, které mohou mikropočítače obsahovat, jsou: ROM, EPROM, EEPROM nebo FLASH, velikost se pohybuje od desítek až po stovky KB
- Vstupně výstupní porty – slouží k příjmu nebo odesílání dat

Kromě základních částí může mikropočítač obsahovat také další periférie, jakou jsou například časovače, řadič přerušení, watchdog timer a další. Mikropočítače jsou také schopné komunikovat s okolním prostředím a to jak s jinými mikrokontroléry tak i se stolními počítači, mohou také ovládat a řídit zařízení jiná.

2.2.1 Mikropočítač ATtiny2313

Tento mikročip od firmy ATMEL je velmi vhodný pro ovládání jednoduchých zařízení. Také se dá považovat za nástupce mikročipu AT90S2313 od téže firmy, oproti kterému přináší několik vylepšení, jako jsou například tři softwarově volitelné pohotovostní režimy nebo vylepšení sériového rozhraní UART na USART, které umožňuje kontrolu přenášených dat. Mikročip ATtiny2313 také obsahuje architekturu, která umožňuje přístup do dvou nezávislých registrů v jednom hodinovém cyklu, což má za důsledek až desetkrát rychlejší zpracování kódu než konvenční CISC mikrokontroléry.

CISC mikrokontroléry obsahují obsáhlou instrukční sadu strojových instrukcí, které mohou být provedeny v různých časových intervalech a také mohou být o různé délce, oproti mikrokontrolérům typu RISC, kde jsou všechny instrukce stejně dlouhé a doba jejich vykonání trvá stejně dlouhou dobu.

Základní parametry mikročipu ATtiny2313:

- 32 registrů
- Napájecí napětí 2.7 – 5.5V
- 2KB programovatelné FLASH paměti
- 128 bytů EEPROM a SRAM paměti
- 3 vstupně/výstupní porty (PA – 3bitový, PB – 8bitový, PC – 8bitový)
- USART
- Watchdog (slouží jako kontrola mikročipu – pokud není Watchdog po předem nastavené době resetován, předpokládá, že se mikročip zaseknul a celý ho resetuje)
- RC Oscilator – udává frekvenci se kterou spíná hodinový signál mikročipu

2.2.2 Základní popis pinů mikroprocesoru ATtiny2313

VCC - napájení mikroprocesoru (5V)

GND - zemnění mikroprocesoru (0V)

PORTA (PA2..PA0) - port A je 3bitový vstupně výstupní port

- na každém bitu portu se nachází jeden pull-up rezistor

PORTB (PB7..PB0) - port B je 8bitový vstupně výstupní port

- na každém bitu portu se nachází jeden pull-up rezistor

PORTB (PD6..PD0) - port B je 7bitový vstupně výstupní port

- na každém bitu portu se nachází jeden pull-up rezistor

RESET - tento pin je tak zvaný resetový vstup

- pokud je na tento pin přiváděno napětí o hodnotě log. 0 déle než 2,5µs, vyvolá se reset mikročipu

XTAL1 - vstup invertujícího oscilátoru a do vnitřního hodinového okruhu

- slouží k připojení externího krystalu

XTAL2 - výstup invertujícího oscilátoru

- slouží k připojení externího krystalu

2.3 Sériová komunikace (USART/UART)

USART je sériové rozhraní, které umí posílat data z mikročipu přes standardní sériovou linku RS232. Oproti staršímu UART je rozšířeno o synchronní režim, což znamená, že oproti původním třem vodičům (zápis, čtení, GND) je potřeba ještě jeden vodič, který slouží jako kontrolní.

2.3.1 Inicializace UART pro mikročip ATtiny2313

Ačkoliv mikroprocesor ATtiny2313 umožňuje sériovou komunikaci USART, vystačil jsem si v této práci pouze s UART, a tudíž budu dále hovořit pouze o tomto typu sériové komunikace.

Před začátkem komunikace mikročipu a PC, musíme nejprve UART inicializovat. Inicializační proces se obvykle skládá z nastavení přenosové rychlosti, sériového rámce a dle využití UART také vysílání nebo příjmu.[4]

Přerušení u UART řídí operace GlobalInterrupt Flag. Ta by měla být při inicializaci vymazána (a přerušení globálně vypnuto). Před provedením re-inicializace, změny Baudrate (rychlost přenosu jednoho bytu) nebo změny formátu sériového rámce se musíme ujistit, že v období provádění změn neexistují žádné probíhající procesy. Pro kontrolu můžeme použít Flag TXC, zjistí, zda vysílač dokončil všechny převody. Pokud chceme také zjistit, zda nejsou ve vyrovnávací paměti (bufferu) ještě nějaká nepřečtená data, použijeme Flag RXC. GlobalInterrupt Flag musí být smazán před každým přenosem (před každým zápisem do URD). [3].

2.3.2 Vysílání dat pomocí UART

Vysílač se zapne povolením nastavení TransmitEnable (TXEN), což je bit v registru UCSRB. Pokud je vysílač povolen, je normální provoz pinu TxD přepsán a předán jako sériový vysílač. Přenosovou rychlost, způsob režimu (read/write) a sériový rámec musejí být nastaveny před provedením jakéhokoliv přenosu. Pokud je užit synchronní přenos, hodiny na pinu XCK musí být přepsány a použity jako přenosové hodiny. [3].

2.3.3 Odesílání rámců o velikosti 5 – 8 bitů

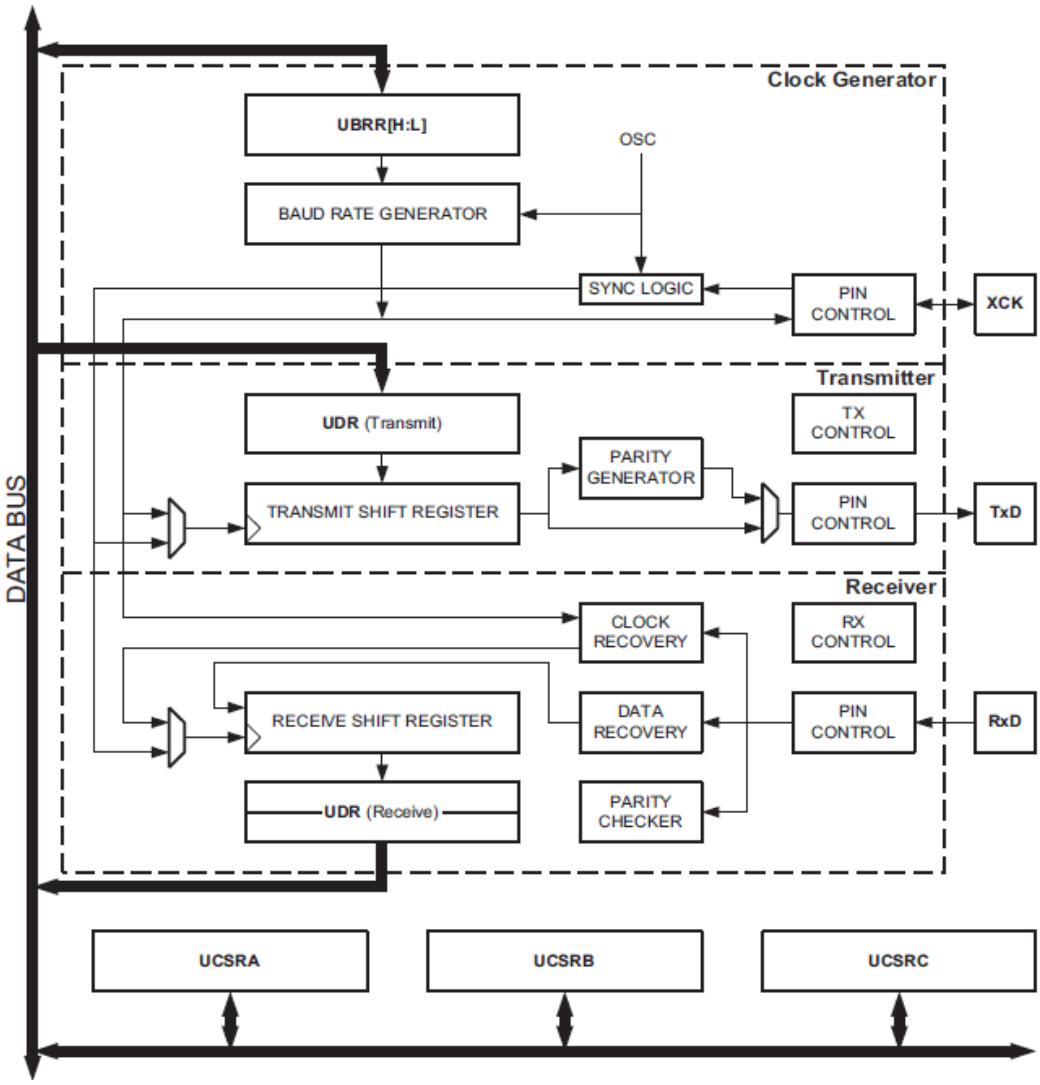
Přenos dat je inicializován tehdy, když do vysílacího bufferu nahrajeme vysílaná data. CPU může načíst vysílací buffer zapsáním do UDR. Data ve vysílacím bufferu budou přesunuta do posuvného registru právě tehdy, když je posuvný registr připravený zaslat nový rámec. Do posuvného registru se data nahrají, pokud je v klidovém stavu (neprobíhá žádný přenos) nebo ihned po posledním stop bitu z předchozího rámce. Když jsou již data v posuvném registru, pošle se kompletní rámec Baud registru. [3].

2.3.4 Příjem dat pomocí UART

Přijímač je povolen zapsáním ReceiveEnable (RXEN) bitu v registru UCSRB na hodnotu jedna. Pokud je přijímač povolen, tak je normální provoz pinu RxD přepsán na sériový vstup přijímače. Přenosová rychlost, způsob provozu a formát rámce musejí být nastaveny před jakýmkoliv příjmem dat. Pokud používáme synchronní režim, musíme hodiny na pinu XCK přepsat a použít jako přenosové hodiny. [3].

2.3.5 Příjem rámců o velikosti 5 – 8 bitů

Přijímač začne přijímat data, když zjistí platný start bit. Každý bit, který následuje za start bitem, bude odebrán Baud (přenosovou) rychlostí nebo rychlostí udávanou XCK hodinami. Po té je poslán do Receive (příjmového) registru, a to se děje do té doby, než je obdržen první stop bit tj. kompletní sériový rámec. Ten je přítomen v příjmovém posuvném registru. Obsah příjmového posuvného registru je přesunut do příjmového bufferu. Tento buffer pak může být přečten pomocí UDR. [3].



Obr. 7 Blokové schéma USART [3].

2.4 Převodník TTL – UART, integrovaný obvod MAX232

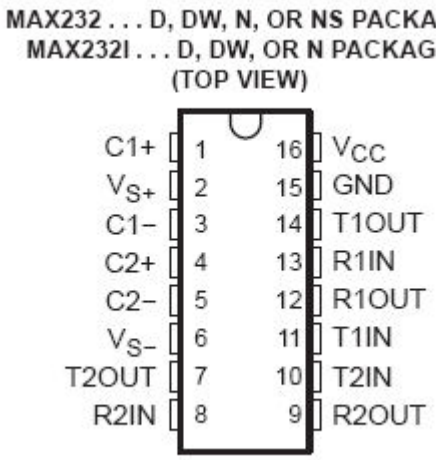
Tento obvod slouží pro převod úrovní z RS232 na TTL a opačně. V případě, že bychom chtěli posílat data pouze mezi dvěma procesory TLL, nemusejí se žádné převody napětí řešit. Ale v případě, že chceme komunikovat mezi logikou TTL a počítačem (RS232), je nutné tento převodník použít.

Musíme tak učinit, jelikož sériové rozhraní RS232 využívá napěťové úrovně ±12V, kdežto logika TTL využívá napětí 0, 5V. Výhodou obvodu MAX232 například oproti převodníku 1488 je to, že odpadá přívod záporného napětí. Toto záporné napětí vyrábí pomocí elektrolytických kondenzátorů a nábojových pump.

Tento obvod také obsahuje dva převodníky napětí z RS232 na TTL a dva převodníky z TTL na RS232.

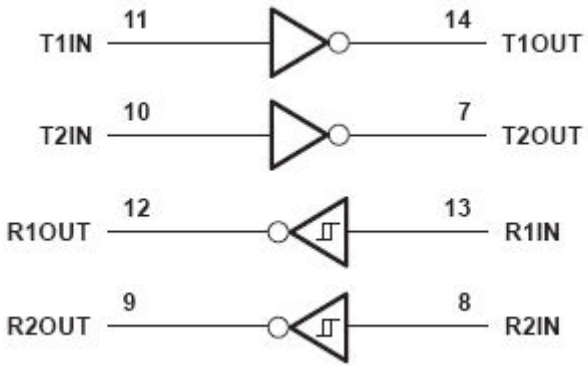
Parametry převodníku:

- napájecí napětí 5V
- 2 napěťové měniče pro každý směr toku
- CMOS konstrukce – vhodná pro baterií napájené obvody

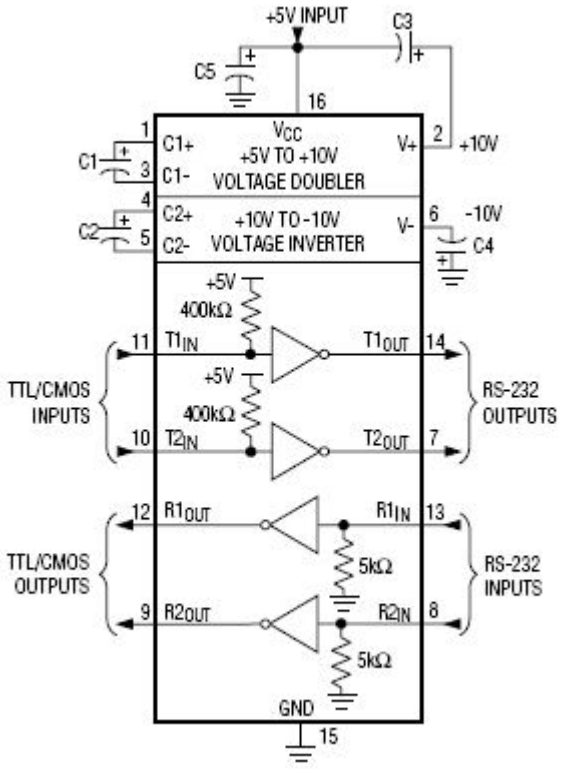


Obr. 8. Popis vývodů integrované obvodu MAX232 [4].

Obvod zvládá operace o rychlosti 120kbitů/s, vstupní úrovně na vstupech RS232 mohou dosahovat úrovní až -30V a odebíraný proud je asi 8mA. Důležité je také podotknout, že obvody neslouží jako prosté budiče, ale jsou to invertory. Viz obrázek 12. [4].



Obr. 9. Zjednodušené vnitřní zapojení datové části integrovaného obvodu MAX232 [4].



Obr. 10. Blokové schéma zapojení integrovaného obvodu MAX232 [4].

2.5 Popis zapojení mikroprocesoru a displeje

Celý obvod je napájen napětím o velikosti 5V. Toto napětí je získáno z obyčejného napájecího zdroje např. Tomy Line SUNNY 5V/3A. Data do mikroprocesoru se nahrávají prostřednictvím paralelního portu, který je k mikroprocesoru připojen následovně:

1. Pin paralelního portu Init je připojen k mikroprocesoru na pin RESET. Init je řídící pin a ovládá jej počítač. Pokud na tento pin bude po dobu delší než 2.5μs přiváděna log. Nula, bude mikročip resetován.

2. Pin paralelního portu Strobe je připojen k mikroprocesoru na pin SCK. Pin Strobe je také řídící a ovládá jej počítač.

3. Pin paralelního portu Busy je připojen k mikroprocesoru na pin MISO. Pin Busy slouží jako stavový a počítač z něj může pouze číst.

4. Pin paralelního portu DO je připojen k mikroprocesoru na pin MOSI. Pin DO je datový.

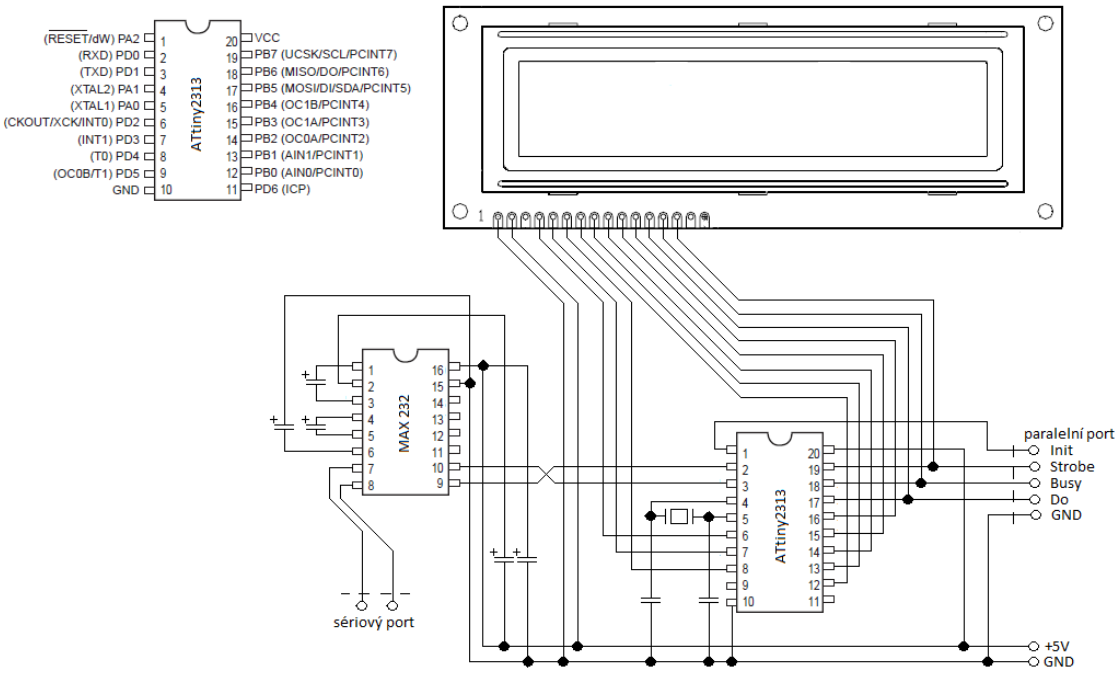
5. Pin paralelního portu GND je připojen na napětí 0V.

Pro komunikaci mikročipu s displejem využívám PORTB a PORTD. V případě portu D jsou potřeba pro komunikaci pouze piny PD2 (řízení signálu RS), PD3 (řízení signálu R/W) a PD4 (řízení signálu E). U portu B je nutno použít všechny dostupné piny PB0 – PB7, z čehož je patrné, že se jedná o 8bitovou paralelní komunikaci. Těmito piny posílám displeji data, která

se vyhodnocují buď jako instrukce, například adresace kurzoru, nebo jako data k zobrazení. Všechny piny portů B i D jsou na displeji připojeny k pinům se stejnými názvy.

Další součástkou připojenou k mikročipu je krystal. Jelikož má mikročip ATtiny2313 již krystal vlastní, není jeho připojení pro samotný chod bezprostředně nutné, ovšem pro sériovou komunikaci je nezbytné. Krystal se připojuje na piny XTAL1 a XTAL2, přes dva 33 pF kondenzátory propojené se zemí.

Poslední součástkou připojenou k mikročipu je integrovaný obvod MAX 232 CPE, tento je připojen na piny RxD (příjem) a TxD (vysílání). Funkce tohoto mikročipu je taková, že převádí příliš vysoké napětí sériového portu (+12 V a -12 V) na napětí snesitelné pro mikročip (5 V).



Obr. 11. Blokové schéma zapojení mikroprocesoru a displeje

Soupis použitých součástek

- mikroprocesor ATtiny2313
- LCD displej EL16023a
- integrovaný obvod MAX232
- krystal 9.216 MHz
- dva kondenzátory 33 pF
- 4 filtrační kondenzátory 100nF
- konektor CANON 25 pin samec
- konektor CANON 9 pin samec

3. PRAKTICKÁ ČÁST

3.1 Program mikroprocesoru

Program mikroprocesoru je napsán v jazyce C a je zkompilován pomocí kompilátoru AVR DUDE. Celý program se skládá ze sedmi souborů: main.c, main.h, lcd.c, lcd.h, uart.c, usart.h a souboru makefile.

Soubor lcd.c obsahuje metody pro komunikaci s displejem. Jednou z hlavních metod je metoda *void init(char mod)*. Tato slouží k inicializaci displeje a je potřeba volat ji před veškerou ostatní komunikací s LCD.

Ukázka zdrojového kódu metody void init(char mod):

```
void init( char mod ){

   //set port for pin E,RS,RW as output
LCD_RS_CR |= ( 1 << LCD_RS_PIN );
LCD_RW_CR |= ( 1 << LCD_RW_PIN );
LCD_E_CR  |= ( 1 << LCD_E_PIN );

   //all pin is 0;
_E(0);_RW(0);_RS(0);

   //set pin as output
LCD_DATA_CR |= ( 1 << PINB0 ) | ( 1 << PINB1 ) | ( 1 << PINB2 ) | ( 1 << PINB3 ) | ( 1 << PINB4 ) | ( 1 << PINB5 ) | ( 1 << PINB6 ) | ( 1 << PINB7 );
   //all pin is 0;

    _delay_ms(60);
    _send_INST(0x30);
    _delay_ms(6);

    _send_INST(0x30);
    _delay_us(150);

    _send_INST(0x30);
    _delay_us(50);

switch(mod)
```

```
    {
     case 0:_send_INST(0x38); break;
     case 1:_send_INST(0x36); break;
    }
    _delay_us(50);

    _send_INST(0x01);
     _delay_ms(2);

    _send_INST(0x0C);
    _delay_us(50);

    _send_INST(0x06);
    _delay_ms(50);

}
```

Na začátku této metody jsou definovány porty B a D. Port D je používán jako řídící, ovládá tedy piny displeje E, RS a RW. Tyto piny je třeba nastavit jako výstupní. Toto se nastaví pomocí registru DDRD, který je reprezentován datovými proměnnými *LCD_RS_CR, LCD_RW_CR a LCD_E_CR* . Po nastavení registru DDRD se nastaví ovládací piny portu na hodnotu 0. Toho docílím, když nastavím registr PORTD daného pinu na nulovou hodnotu, např.: *LCD_E_PORT &= ~(1 <<LCD_E_PIN).*Toto vykonávají metody *_E(0);_RW(0);_RS(0);*.

Dále je potřeba nastavit port B, který bude pro inicializaci opět funkčně nastaven jako výstupní. V jeho případě se použijí všechny piny portu, proto celý registr DDRB nastavím na hodnotu 1.

Zbylé řádky metody slouží k samotné inicializaci displeje, která je popsána v kapitole *2.1.3.* K odesílání inicializačních instrukcí displeji je používána metoda *_send_INST(charval),* která využívá k odesílání instrukcí metodu _send_DI(char val , char R_S).

Metoda _send_DI(charval, char R_S) již přímo komunikuje s displejem a odesílá na jeho sběrnici požadovaná data nebo instrukce.

Ukázka zdrojového kódu metody _send_DI(charval , char R_S):

```
void _send_DI( charval , char R_S )
{
   _RS(R_S);
_delay_us(4);
   _RW(0);
_delay_us(4);
   LCD_DATA = ( val );
_delay_us(4);
_E(1);
   _delay_us(4);
_E(0);
   _delay_us(4);
}
```

Prvním argumentem, který tato metoda přijme je hodnota, která má být na displej odeslána. Tato hodnota se pomocí *LCD_DATA = (val)* nastaví na piny portu B. Druhým přijímaným argumentem je hodnota RS. Ta je ihned přidána metodě *_RS(char val),* ta nastaví řídící pin portu D a rozliší, zda se jedná o instrukci či data.

Po nastavení řídících hodnot RS, RW a nastavení hodnot pinů portu B, se povolí přistup zápisu dat na displej pomocí metody *_E(chat val).* Po časové prodlevě 4μs jsou data korektně zapsána na displej a přístup je potřeba opět zamezit.

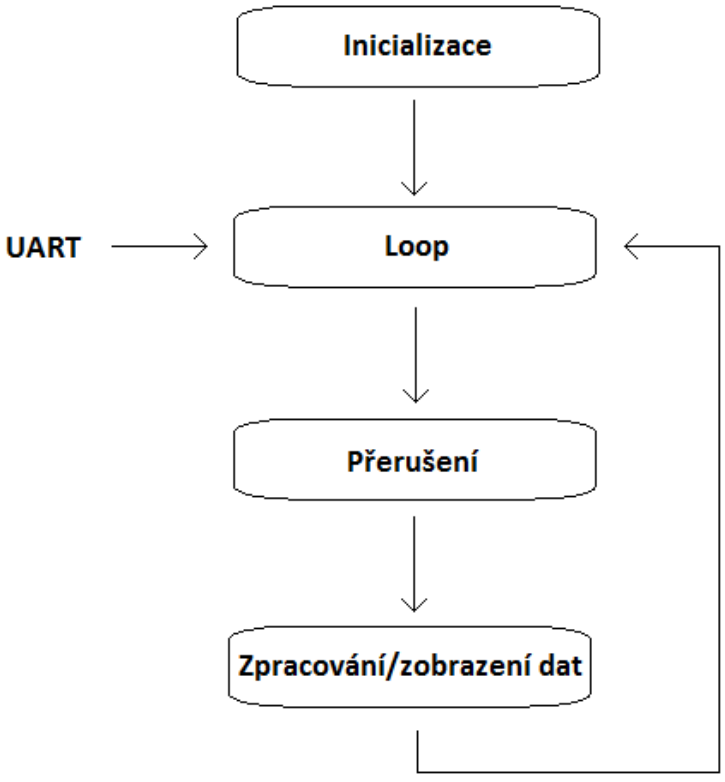
Pro vykreslení jednoho bodu v grafickém režimu slouží v souboru lcd.c *funkce void point(char radek, char sloupec).* Každý bod musí být vykreslen tak, aby neovlivňoval jiný potenciální bod ve stejné 16bitové buňce. Proto je potřeba nejprve obsah této buňky přečíst a získanou hodnotu logicky vynásobit s hodnotou zapisovaného bodu. Abych tohoto docílil, musím nejprve zaslat na displej instrukci s požadovanou adresou a řídícím pinem RW nastaveným na hodnotu jedna. Tím displeji sdělím, že chci obsah této adresy přečíst. Po zaslání dotazu na adresu je nutné přenastavit datový port mikročipu jako vstupní. Toho se docílí nastavením pinů registru DDRB na hodnotu log. 0 a nastavením pinů registru PORTB na log. 1, čímž se aktivují pull-up rezistory. Nyní se mohou zaslat displeji řídící instrukci E. Jedním aktivováním a následným deaktivováním řídícího pinu E se provede tak zvaný „*dummyread*“. Druhou a třetí aktivací a následnou deaktivací řídícího pinu E již dostáváme od displeje data. Tyto data lze číst na pinech registru PINB. Po přečtení těchto dat opět nastavím port B jako výstupní a vykreslím bod.

Soubor uart.c obsahuje nastavení přenosové rychlosti baud rate na hodnotu 9600, a také nastavení registru UCSRB ve kterém se přepisují standardní piny portu D na piny pro sériovou komunikaci. Registrem UCSRC nastavuji přímo komunikaci a to bitem USBS, který nastavuje stop bit a bitem UCSZ0, kterým se nastaví velikost přenášeného rámce.

V souboru main.c je nejprve nadefinován sériový přenos dat (UART), dále je v něm volána inicializace displeje a hlavně jsou v něm také metody sloužící pro přerušení a odesílání dat pomocí UART. Po nastavení sériového přenosu a inicializaci displeje se program dostane do nekonečného cyklu *while(1),* ve kterém čeká na příchod dat do UART. Jakmile nějaká data dorazí, dojde k přerušení a zobrazení či vyhodnocení přijatých dat.

Soubor makefile slouží k řádné kompilaci všech souborů. Z těchto zmíněných souboru vytvoří jeden binární soubor typu INTEL HEX, který následně nahraje do paměti FLASH mikročipu. Soubor makefile obsahuje další funkce jako je vymazání nepotřebných souborů, které makefile při kompilaci vytvoří a v neposlední řadě také obsahuje funkci, která deaktivuje vnitřní krystal procesoru. Tato funkce nesmí být použita bez připojení vnějšího krystalu, protože by procesor přestal fungovat, nic by mu nekmitalo s vnitřními hodinami. Jelikož je ale vnitřní krystal mikročipu nevhodný pro sériový přenos dat, je nutné externí krystal využívat.

Zbývajícími soubory jsou main.h, lcd.h a uart.h, tyto soubory slouží k vytvoření statických knihoven.



Obr. 12. Diagram programu mikroprocesoru

3.2 Aplikace pro OS Linux

Tato aplikace slouží uživateli k obsluze LCD displeje pomocí standardního sériového rozhraní RS232. Je rovněž, jako program mikroprocesoru, napsána v jazyce C, ale je již zkompilována klasickým unixovým kompilátorem GCC. Celá aplikace se skládá pouze z jednoho souboru program.c a knihovny pro sériový port rs232.a.V knihovně pro sériový port se nacházejí funkce, sloužící k otevření a nastavení sériového portu, také zde najdeme funkce pro odesílání a příjem dat.

Aplikace umožňuje práci s displejem v textovém i grafickém režimu. Ihned po spuštění se program nachází v grafickém režimu, ze kterého lze ovšem velmi snadno přejít do textového. Menu programu je tedy rozloženo do dvou metod.

Smyčka pro ovládání grafického režimu je napsána přímo v hlavní metodě *void main()*. Z této smyčky se pak program dále odkazuje na požadované funkce do jiných metod. Ovládání textového režimu je velmi podobné režimu grafickému, pouze s tím rozdílem, že ovládací smyčka je uložena v metodě *void textMode()*. Mezi těmito režimy a jejich funkcemi můžeme při práci s displejem libovolně přepínat.

Ukázka zdrojového kódu metody intopen_port(void) z knihovny rs232.a:

```
Int open_port(void)
{
int fd;

fd = open("/dev/ttyS0", O_RDWR | O_NOCTTY | O_NDELAY);

if(fd == -1)
  {
perror("open_port: Unable to open /dev/ttyS0 - ");
  }
else
  {
fcntl(fd, F_SETFL, 0);
  }

return(fd);
}
```

Metodou *open()* se přistupuje k souboru sériového portu, který je uložen ve složce *dev* a jeho název je *ttyS0.* Nula na konci jeho názvu značí, o který sériový port se jedná, v počítači může být sériových portů více, pokud bychom chtěli využít jiný port než ttyS0, museli bychom zadat cestu ttyS1 apod. Dále se v metodě *open()* kromě cesty nastaví i přístup k danému souboru, čtení/zápis a jak se má tento soubor chovat.

Pokud se nám podaří daný soubor korektně otevřít, použijeme na něj funkci *int fcntl (int *filedes, intcmd, ... /* intarg */),*která slouží k nastavení deskriptoru. V tomto případě pomocí *F_SETFL*k nastavení příznaků deskriptoru. Aby funkce věděla, ke kterému deskriptoru má přistupovat, musíme jí deskriptor předat v jejím prvním argumentu.

Po úspěšném otevření a nastavení souboru sériového portu, je ještě potřeba korektní nastavení přenosu dat. K tomu slouží metoda *int configure_port(int fd).*

Ukázka zdrojového kódu metody intopen_port(void) z knihovny rs232.a:

```
Int configure_port(int fd)
{
struct termiosport_settings;

cfsetispeed(&port_settings, B9600);
cfsetospeed(&port_settings, B9600);

port_settings.c_cflag&= ~PARENB;
port_settings.c_cflag&= ~CSTOPB;
port_settings.c_cflag&= ~CSIZE;
port_settings.c_cflag |= CS8;

tcsetattr(fd, TCSANOW, &port_settings);
return(fd);

}
```

V této metodě se nastavuje přenosová rychlost baud rate, která musí být stejná jako v mikročipu. Dalším nastavením v této metodě je *port_settings.c_cflag&= ~PARENB – zrušení paritního bitu, port_settings.c_cflag&= ~CSTOPB – zrušení stop bitu,port_settings.c_cflag&= ~CSIZE – zrušení datového bitu aport_settings.c_cflag |= CS8 – nastavení přenášeného rámce na 8bitů.* K samotnému zápisu dat na linku RS232 je využita funkce write().

Soubor program.c obsahuje hlavní funkci *void main(),* ve které se volají funkce z knihovny rs232. Kromě volaných funkcí pro nastavení sériového rozhraní obsahuje funkce main také

hlavní smyčku, do které se uživatel dostane ihned po spuštění programu, tato smyčka reprezentuje hlavní nabídku programu. Uživatel si v jednoduchém menu vybere požadovanou funkci tím, že načte ze standardního vstupu hodnotu, která tuto funkci reprezentuje. Může si vybrat mezi funkcemi pro zobrazení nápovědy, vykreslení jednoho bodu, úsečky, kružnice, vymazání displeje, přepnutí do textového režimu nebo ukončení programu.

Volba nápovědy obsahuje soupis všech dostupných funkcí a jejich jednotlivé zkratky, pomocí kterých v menu program ovládáme. Pro přehlednost a snadné ovládání to je vždy první písmeno volby, kruh – k apod.

Pokud uživatel zvolí kteroukoliv z geometrických funkcí (bod, úsečka, kruh), je vždy programem vyzván k zadání souřadnic, na kterých se má daný objekt vykreslit. Tyto hodnoty se načítají ze standardního vstupu a odesílají skrze sériové rozhraní RS232 do mikročipu. Ten tyto data ve svém interním programu zpracuje a pomocí displeje zobrazí.

Pro výpočet jednotlivých bodů úsečky jsem využil Bresenhamova algoritmu. Ten spočívá v tom, že se nejprve zjistí delší a kratší strana obdélníku, ve kterém je úsečka vykreslena a podílem kratší strany s delší získáme směrnici přímky. Tuto směrnici pak již pouze vynásobíme s iterační hodnotou, která se v každém cyklu inkrementuje o jedna, a připočteme výchozí souřadnici. Takovýto výsledek je ovšem velmi často s přesností na několik desetinných míst, proto jej ještě musíme zaokrouhlit na celá čísla. Tímto získáme souřadnici úsečky, která leží na ose kratší strany obdélníku. Souřadnici, ležící na ose delší strany obdélníku pouze v každém cyklu inkrementujeme o jedna od počáteční hodnoty.

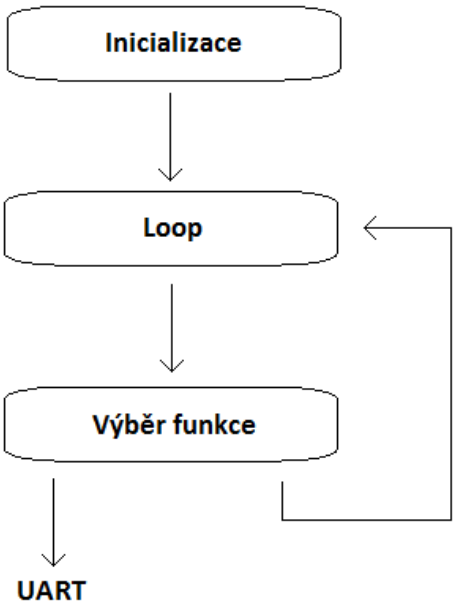
Ukázka zdrojového kódu metody void drawLine(int ay, int ax, int by, int bx):

```
...
for(i = 0;i<longLine+1;i++)
    {
      if(longLine == stranaX)
      {
         is = i*smernice;
         pom3 = (int)is;
         des = is - pom3;
```

```
        des = roundi(des);

        x = ax + i;

        y = ay + des + pom3;

        setPoint(x, y);

    }
```

V této části zdrojového kódu se provádí již zmíněný výpočet souřadnic přímky. Cyklus *for(i = 0;i<longLine+1;i++)* bude probíhat tak dlouho, dokud hodnota i nenabude stejné hodnoty jako je délka delší strany, díky tomu se úsečka vykreslí přesně o takové velikosti, jakou požadujeme. Po té si v kódu ujasníme, která strana obdélníku, ve kterém se úsečka vykresluje, je delší. Toto se provede podmínkou *if(longLine == stranaX)*, pokud je *stranaX* opravdu rovna hodnotě *longLine*, víme, že výpočet souřadnice bodu pomocí směrnice přímky použijeme pro osu y a osu x budeme pouze inkrementovat o hodnotu jedna. Dalších několik řádků pouze provádí již zmíněný výpočet a pomocí funkce *void raundi(double a)* se provede ono nezbytné zaokrouhlení. Nakonec výsledné souřadnice pouze odešleme skrze sériové rozhraní RS232 mikročipu pomocí metody *void setPoint(int x, int y).*

Metoda *void setPoint(int x, int y)* dělá pouze to, že nejprve odešle znak ¨p¨, kterým mikročipu sdělí, že se bude vykreslovat bod. Následně už jen pošle hodnoty x a y, které reprezentují hodnoty souřadnic, ve kterých se bod na displeji zobrazí.



Obr. 13. Diagram aplikace pro OS Linux

4. ZÁVĚR

Moje bakalářská práce se skládá z dvou částí. První část je teoretická a je v ní veškerý popis mnou použitých součástí, výčet jejich funkcí a podrobný popis práce s nimi. Veškerá literatura, kterou jsem v této části využíval je sepsána v kapitole 5 Literatura.

Druhá část mé práce je praktická a zahrnuje mnou vytvořený program jak pro mikročip, tak i aplikaci pro OS Linux. Při programování těchto dvou aplikací jsem se setkal s problémy zejména u programu pro mikroprocesor. Uváděné informace v datasheetech nebyli vždy úplně korektní. V průběhu vytváření mého výrobku a následném programování jsem zjistil, že tato práce přináší dvakrát více problémů, než jsem původně čekal. Jelikož jsem místo leptání a použití kuprextitové destičky využil pájivého pole, velmi často jsem se setkával s problémy nejen algoritmickými, ale také s mechanickými, zejména po přenosu.

5. LITERATURA

[1] Matoušek, David. *Práce s inteligentními displeji LCD, 1. díl.* 1. vyd. Praha: BEN – technická literatura, 2006, 222s. ISBN 80-7300-121-7

[2] Sitronix Corporation. LCD Module ST7920. [online].
URL: <http://www.crystalfontz.com/controllers/ST7920.pdf>[cit. 2008-08-18].

[3] AtmelCorporation. 8-bitMicrocontroller Attiny2313.[online].
URL: <http://www.atmel.com/dyn/resources/prod_documents/doc2543.pdf>
[cit. 2010-08-10].

[4] 8bitu.cz. [online].
URL: <http://www.8bitu.cz/clanek/obvod-max232/>[cit. 2007-01-07].

[5] abclinuxu.cz. [online].
URL: <http://www.abclinuxu.cz/clanky/hardware/jednocipy-pod-linuxem-i>
[cit. 2003-18-12].

[6] wikipedia.org. [online].
URL: <http://en.wikipedia.org/wiki/Midpoint_circle_algorithm>[cit. 2011-1-8].

[7] wikipedia.org. [online].

URL: < http://en.wikipedia.org/wiki/Bresenham's_line_algorithm>
[cit. 2011-29-7].

6. PŘÍLOHY

Příloha 1 - Program pro mikroprocesor ATtiny2313
Příloha 2 - Obslužná aplikace mikroprocesoru ATtiny2313 pro OS Linux